AUSTRALIAN OS9 NEWSLETTER

EDITOR :
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

(07) 345-5141

MAY 1990

ATTR Syntax: Attr filename [permissions] Usage : Examine or
change the security permissions of a file Opts: -perm = turn off
specified permission perm= turn on specified permission -a =
inhibi rms : d - directory file
s - no to owner w - write permit
to own r - read permit to public
pw - te permit to public BACKUP
Synta ge : Copies all data from
one de ead error occurs s =
single writes BASIC09 Syntax :
Basic0 ge BUILD Syntax: Build
filena s from standard input
CHD S nge working directory to
specifi > Usage: Change execution
directory to specified path CMP Syntax: Cmp filename1 filename2
Usage : File comparison utility COBBLER Syntax: Cobbler devname
Usage : Creates OS-9 bootstrap file from current boot CONFIG
Syntax: n disks COPY
Syntax data from
one fil E Syntax :
Date [t Opts: t =
specify ame> Usage
: Check directory
for wor sters -m
= save of unused
cluster nly -o =
print <devname>
]<devn : Del [-x]
filenam s : -x =
delete x: Deldir
directo yntax: Dir
[e] [x] the file
names y x=print
executi] Usage :
Display s converted characters to standard output DSAVE Syntax
: Dsave [-opts] [dev] [pathname] Usage : Generates procedure file
to copy all files in a directory system Opts : -b make a system
disk by using OS9boot if present -b=<path> = make system disk
using path do not
process b makdir
command o num K
ECHO Syn tandard
output ED oriented
text edito s text
error messages for given error numbers EX Syntax: ex <modname>
Usage: Chain to the given module FORMAT Syntax : Format
<devname> Usage : Initializes an OS-9 diskette Opts ; R - Ready L
- Logical format only "disk name" 1/2 number of sides 'No of

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 4 Number 4

EDITOR : Gordon Bentzen

HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 User Group.

Welcome to the May edition of our monthly newsletter. We again appeal for your comments and for submissions to be included in future editions as this newsletter is intended as the newsletter of the OS9 user group. We continue to try to present articles which will be useful to OS9'ers at all levels of experience.

I have had a few queries over the last couple of months from members who have just begun the venture into OS9 and predictably are having some difficulties, so I have decided to commence a series, "OS-9 From The Beginning", which is aimed at the new OS9 user. Bob Devries often reminds me that "it is all in the manual" when I ask a question at our local user group meeting. Whilst this is no doubt true, I seem to avoid reading the manual unless it is almost a last resort. The OS9 manual after all can be a little daunting to begin with, so I hope to guide the new user through the learning curve with several references to this manual.

Bob Devries had a call from Mike Haaland, author of MVCanvas, of Las Vegas, Nevada, this week and gleaned some more first hand information about the KLE (Kenneth-Leigh Enterprises) 'CoCo4'. It has a 68070 (Signetics) CPU, 2 MByte RAM, 1.44 MB 3.5 inch drive, and OS9/68000 kernel in ROM. Mike Haaland is one of the software developers for the new machine, so we should see some nice software for it. The new machine is already being produced, and the first batch will be given to software developers only, and the computers will be available for purchase in August this year.

Mike also mentioned that version 2 of MVCanvas is ready for shipping, and should be available from Nickolas Marentes by the time you read this. Purchasers of version 1 will be able to buy an upgrade for approximately $15.00, and the new price will probably be $50.00. Contact Nickolas Marentes for more details. Also read the review in the March '90 issue.

An interesting opportunity came to Don Berrie recently, when he was able to get limited access to a mainframe computer which had access to Bitnet, a global information network. It is this network which has provided the material about the new computers which we have published recently. Those files, of course were kindly provided by Peter Edwards, one of our members from Melbourne. Also available on this network, are a large number of public domain and shareware programmes, but in a strangely encoded form to make them transmittable as textfile-like electronic mail.

Don has managed to 'download' some of these files, and is in the process of decoding them. In due course, they will be made available as part of the National User Group public domain collection. The usual conditions for copying will apply.

One of the things which Don has downloaded, is an archive which provides patches to both the GRFDRV and GSHELL+ modules, to enable them to run faster. When installed, these patches really make the Multiview environment fly. In fact, Don has timed multiview drawing a screen full of icons (form his commands directory) at 1.2 seconds. This compares with 2.8 or so seconds with the unpatched versions. There is really still some life left in the old 6809 yet!!

Cheers, Gordon.

Documentation for ZAP.

The following programme is what could be described as a 'home-made' disk editor written expressly for the Color Computer 3 and the OS9 Level II operating system. Copying by any method, including posting on electronic Bulletin Board Systems, for personal or educational use is encouraged. Copying for commercial purposes, or for sale or resale is expressly forbidden.

The following equipment and software utilities (which may be the proprietory property of other software/hardware vendors) are required to successfully operate the programme:

HARDWARE

1. 512K Tandy Color Computer 3
2. OS9 Level II Operating System
3. High Resolution Monitor

(The use of a colour monitor is advantageous, but not essential)

SOFTWARE

The programme needs, at various times during it's execution, access to the following system modules or commands:

RunB, Syscall, gfx2, shell, xmode, dir, /P or /P1 and /W
Syscall needs to be merged with either Gfx2 or RunB to allow RunB to find it.

The programme requires an unused window descriptor in which to run itself.

To use the programme, copy it to your current execution directory, and at the OS9 prompt, type ZAP.

```
-WARNING--WARNING--WARNING--WARNING--WARNING--WARNING--WARNING--WARNING-
:                                                                      :
W    This programme performs low  level read  and write  operations    W
A    on RBF devices.  Use with  extreme care.   Do not use on media    A
R    which  is  irreplaceable.  If  you  do  not  understand what a    R
N    programme of  this  nature is used for,  do not attempt to use    N
I    it.  Sector copies, including those from one device to another    I
N    are  permitted.   However  use  extreme  care,  as  the sector    N
G    allocation  tables  are not  updated  during  this  operation.    G
:                                                                      :
-WARNING--WARNING--WARNING--WARNING--WARNING--WARNING--WARNING--WARNING-
```

This documentation should accompany all copies of the programme, so that users have access to the address of the authors for advice and comments. This project has been supported by the Australian National OS9 Users Group.

PROGRAMME USAGE

The programme is started from the command line by simply typing "ZAP" with no arguments. The introductory banner is then displayed, together with a warning, and the user is prompted to input a device name. The device name entered is checked for validity, and if valid, an attempt is made to read LSN0 of the media currently associated with that device.

If LSN0 can be read, the media size bytes are decoded, and the user is prompted to enter a sector number. The programme will not accept sector numbers larger than the maximum number read from the media or less than zero. As soon as an acceptable sector number is input, that sector is output to the screen in a matrix comprised of 16 rows by 16 two-character bytes, and a menu is presented at the bottom of the screen.

The following list describes the menu items in alphabetical order.

A - Toggles on and off an ascii decoding of the current sector. Only printable ascii characters are printed. Non-printable characters are represented by a '.' character. This ascii mode remains in force until a further 'A' is entered to turn it off.

C - Switches on a simple calculator in an overlay window. The calculator only accepts input of the following keys. 0 - 9, A - F (shifted or unshifted), the + (;) - * (:) / and <ENTER> keys. The user may optionally use the <SHIFT> key for + and *. The calculator always expects input in hexadecimal mode, however the decimal equivalent of the calculator is displayed in the bottom half of the overlay window. To exit from the calculator mode (at any time, even in the middle of the calculation) simply press the <F1> key.

D - Prompts for a change of current device. If the programme cannot open the new device, or attempts to open a device with a media error, it will terminate and display the OS9 error number which caused the problem. When a new device is selected, LSN0 is read, and the programme again prompts for a valid sector number.

E - Provides access to the 'dir' command to do an extended directory of a chosen device. The user is prompted to supply a (full) pathname to the directory required.

H - The help command provides the user with an overlay window that shows a (greatly simplified) summary of all of the meanings of the individual menu keys.

I - Shows, in an overlay window, the plain language decoding of LSN0 for the currently selected media in the current device.

M - WARNING! The 'M' command is used to modify the contents of the current sector. These modifications are performed initially in memory. When this command is selected, a new menu is shown, and a cursor appears on the data area of the screen. The cursor is steered around the data area by the use of the arrow keys. It operates on the 'closed system' principle. That is, if the left arrow key is pressed when the cursor is located at byte 0, the cursor will jump to byte 255 and vice-versa. The 'W' key is used to rewrite the current sector, and the user is prompted to verify the operation. Pressing 'H' while in modify mode will provide a simplified help screen. The sector modification routine only accepts pairs of hexadecimal numbers, and moves to the next adjacent position after each pair of numbers is entered. If the ascii switch is toggled on, changes to the sector are NOT shown in ascii until after the new data is written to the media. It is possible to change any data in any sector on the disk. This may make your disk unreadable, or destroy a directory or file, or cause an executable file to have a CRC error. Be careful!!

N - The 'N' key is used to select a new sector on the current media. Only valid sector numbers are accepted. When a valid sector number is entered, the sector data is redrawn to show the contents of the new sector.

P - To print a hard copy of the current sector use the 'P' command. The programme will look for the device /P1 to use to print the sector data. If /P1 cannot be found, the programme will then attempt to use the device /P. If the printer is connected, but offline, the user is prompted to place the printer online, and to retry. If no printer is connected, then (because of a hardware feature of the CoCo 3) the programme appears to see a printer and passes the data to the printer port. As soon as all the data is passed, the programme is ready to accept further input.

Q - Press the 'Q' key to exit the programme. On a successful exit, all windows and devices are closed, all of the programme modules removed from memory, and the user is returned to the original screen from which the programme was run. In most cases, pressing the <BREAK> key will also terminate the programme without messing-up the system.

S - WARNING! This command allows sectors to be copied, either from sector to sector on the same disk, or to media in another device. Be very careful when using this command. The sector allocation table is NOT updated by this operation. The user is prompted for verification before performing a low-level write. This operation may overwrite some of the vital contents of your disk, and make part or all of the disk unusable!! Be very careful.

<= => - The left and right arrow keys are used to move to the next or previous sectors. When located at sector zero, pressing the left arrow will cause the highest numbered sector on

the disk to be displayed. Similarly, pressing the right arrow key when located at the maximum sector causes sector zero to be displayed. If the ascii toggle in on, the new ascii data is written after each sector change.

When expecting input from the users console, the programme will accept both lower case and upper case letters. If the programme terminates with an error condition, an attempt will be made to identify the particular error which caused the termination. It may be necessary to press the escape key to finally return to the OS9 prompt in these circumstances. Extensive error trapping has been incorporated into the programme, however, so as to make as few assumptions as possible about a user's particular system, we have decided not to try to use error number decoding (by looking for /dd/SYS/errmsg etc).

The programme has undergone a number of substantial revisions since the Basic09 source code for version 1.10 was published in the Australian OS9 Newsletter, and a number of major problems have been rectified. The programme will work equally well with hard, floppy, or ram disks of any size, and all necessary disk data is always read directly from the media in question. Extensive debugging and testing has taken place, and as far as we know the programme has no major bugs. Further development is contemplated, however new releases will be written in 'C', and will be of a more generic nature.

If you feel that this programme will be useful, and will help get you out of trouble, please send $10.00 to give us the incentive to keep support for ZAP going, and to keep working on new versions. Feedback from users is encouraged, and may be directed to :

Don Berrie
25 Irwin Terrace
OXLEY Qld 4075
AUSTRALIA

(07) 3753236

Bob Devries
21 Virgo Street
INALA Qld 4077
AUSTRALIA

(07) 3727816

OS-9 FROM THE BEGINNING

We have presented a number of articles in our newsletters designed to help others gain an understanding of this operating system, refer to the Newsletter Index in last month's edition.

For this series I will assume the reader has a Tandy CoCo3 and the Tandy OS-9 Level 2 operating system. I will also assume that the reader may have ventured into the Microware OS-9 package only to decide that it seems very unfriendly and difficult to understand. Perhaps it is time to retrieve the two OS-9 disks from the disk box and the manual from the bottom drawer.

I have used a Tandy 1, 2 and 3, and like many others, became reasonably familiar with the built in Basic, Extended Basic and then Disk Basic. We can run many commercial programs, and even write our own in Basic or assembly language, load files and save files using floppy disks and do many other interesting things; so why OS-9?

OS-9 is an operating system, unlike Disk Basic (sometimes referred to as RS-DOS) which is strictly not an operating system at all, so the very first thing to do is to forget all the Disk Basic commands because the disk access commands are only an extension of BASIC. The one command exception is "DOS" which is a disk basic command used to start-up OS-9.

The Microware OS-9 operating system, once booted, will replace Disk Basic in the computer's memory. All descriptions of hardware devices, eg. disk drives, serial port, etc, which were "built in" to RS-DOS, are now gone. So for OS-9 to use the hardware devices, it needs a description of each and every device connected to the system.

OS-9 has been modelled on UNIX, has a great similarity to UNIX, and will bring a UNIX feel to your favourite computer. This is mainly due to the common concept of "Modules" and unified I/O (Input/Output). OS-9 is made up of over one hundred small programs or utilities, each designed for one small job and referred to as a "Module". To add a new device to OS-9, just describe it to the operating system. Each device needs a "device driver" and "device descriptor". While each separate device needs its own "descriptor", like devices are able to share the same "driver", eg CC3Disk is the "driver" for floppy drives, and D0, D1, D2 are the "descriptors" for each physical drive which each share the CC3Disk driver. OS-9 also uses special class drivers. In OS-9 jargon, a floppy disk or hard disk fall into the special class called "Random Block File" (RBF) devices. However, more on that at a later stage.

The point being that the modular concept of OS-9 allows EACH device to have a unique description. Drive 0 (D0) may be 35 track single sided, Drive 1 (D1) may be 40 track double sided, Drive 2 (D2) may be 80 track double sided, or in fact any combination desired. Each such device must of course have a "descriptor" which matches the physical device.

FIRST, take the OS-9 manual and read the first section "Getting Started". Now don't be tempted to go further into the next sections, but read "Getting Started" again. Don't worry if it does not all make sense at this point.

The "Getting Started" section lists hardware requirements as a CoCo3 (presumably 128K) and one floppy drive. OS-9 on such a system is yeeerrry limited and painful at best. A 512K CoCo3 with two Double sided drives, 40 or 80 track is really needed. To boot OS-9 from the disks distributed by Tandy (35 track, single sided format) a 35 tr. or 40 tr. single or double sided drive, as Drive 0 (D0) will be needed. CoCo/OS9 will support three floppy drives. The disk controller imposes this limit, not OS-9.

The terms "boot" or "bootstrap", by a long standing convention, are simply the terms used for starting up an operating system on a computer.

After only a little use of OS-9 it will be noted that the system is very disk intensive. That is, the system accesses the disk drives frequently for the commands and files it needs to perform the requested tasks. Frequently used command modules can be loaded into memory to reduce disk access and speed things up, but, once again, more on that later.

An RSDOS disk has only one directory (stored on track 17) and all files stored on the disk are recorded in this directory, while an OS-9 disk can have many directories and many levels. "Levels" there is a new concept. OS-9 uses a hierarchal directory system, and it is very important to understand this concept. Much like a family tree or sometimes described as an upside down tree with its "root" at the top. To get to any branch of the tree we must start from the "root"

The Root directory on a disk could be called the highest level, or referred to as the first level accessed. It does not have a name as such. To see the contents of the "root" directory of a disk in drive 0, type "dir /d0", if the disk were in drive 1, type "dir /d1". The root is the first directory read by the device.

The next most important concept to understand about disk files is the way OS-9 uses TWO directories in its operation the DATA directory and the EXECUTION directory. When OS-9 is booted, the data directory defaults to /d0 and the execution directory defaults to /d0/CMDS (this assumes that OS-9 is booted from the floppy drive 0). The execution directory "CMDS" contains executable modules which may be system utilities or program modules. When a command is entered at the OS9: prompt, OS-9 will first look in memory for a module name which matches and if no match is found it will look in the current execution directory and lastly will look in the current data directory. Once OS-9 finds the module, it will attempt to execute that module if in memory, or will load the module from disk and then execute it.

Perhaps the most common error reported by OS-9 is ERROR 216 (Path Name Not Found).
This error may mean that the file or "module" that has been asked for, either from the keyboard or another program, does not exist, OR, more commonly, the file does not exist in the current DATA or EXECUTION directories.

Now the DATA and EXECUTION directories need not remain pointed to /d0 and /d0/CMDS as these can be changed to any valid (existing) directory. eg. chd /d0/CMDS would change the current DATA directory to the CMDS directory on Drive 0. Yes, the data and execution directories can be pointed to the same directory. The DATA directory is also often referred to as the WORKING directory (the same directory by a different name) The command PWD or pwd will report the current WORKING or DATA directory.

What is a "pathname"? Files or modules which are not in the current directories may be located by OS-9 if the correct "pathname" is given. "Pathname" is simply the "path" in the directory or tree structure. eg. the command line list /d0/SYS/errmsg would work provided the file "errmsg" was in the directory SYS which is at "root" level on the disk in D0.

OS-9 assumes that any filename entered at the OS9: prompt is an executable module. It first looks in memory for a module of that name, if not found, it will look in the current execution directory, and if not found, it will look in the current data directory, if not found, will report ERROR #216.

Notice that OS-9 is not "case sensitive", commands entered in lower case are treated identically to commands entered in upper case. This also applies to file names and directory names, however by convention, upper case should be used for all directory and sub-directory names, and lower case used in names of all single files within a directory. This convention makes it easy to distinguish between directories and files. eg The root directory of disk 1 of the level 2 distribution disk looks like this :-

```
 Directory of /d0
OS9Boot        CMDS           SYS            startup
window.t38s    window.t80s    window.glr4    makecopy
```

With the exception of "OS9Boot", a special OS-9 file, we can tell at a glance that "CMDS" and "SYS" are the only two directories at the "root" level. Each of the others is a single file.

OK. "Getting Started" Chapter 2 (pages 2.1 - 2.5) gives all the necessary instructions to "boot" OS-9. At the "Time?" prompt enter the date and time, do NOT just hit <Enter> as the time is important to the operating system. One short cut is to leave out the delimiters. eg 9004012130 works just the same as 90/04/01 21:30. Two digits must be used for each item. 04 for the fourth month, NOT 4.

Assuming that OS-9 has booted O.K. we now have the operating system running with the default (as distributed) hardware descriptions. If this does not suit the capabilities of hardware connected to the system we must set about correctly describing the "devices" to OS-9. We would also like to make these descriptions permanent so that when the operating system is booted, the correct descriptions will already be in place.

There are at least three ways to do this, but perhaps the easiest to begin with is the use of "Config", although it does impose some limitations.

"Getting Started" Chapter 7 (from page 7.1) gives all the detailed instructions. The floppy drive descriptors to match the MAXIMUM capabilities of the system's drives should be selected. OS-9 is smart enough to read and write to floppy disks of any lesser capacity. For example, the descriptor for an 80 track double sided drive as Drive 1 would be D1_80D. Once the system is booted with this descriptor included in the OS9Boot file, Drive 1 would happily read and write to a disk formatted as 40 track double sided, single sided OR 35 track double sided, single sided.

Do not include the descriptor T1 (it won't work very well on the CoCo3). T3 is probably not much use either, so don't include this one. M1 and M2 are for the Tandy Modem Pak, not sold in Australia to my knowledge, so leave these out also. (Note, the Tandy RS232 Pak is not the same as the Modem Pak.)

Include all the window descriptors W, W1 - W7 (512k system assumed) and a [F]ULL command set. Config will create a bootfile, OS9Boot even if [N]O commands are selected, however as a minimum, for a bootable disk it must contain a CMDS directory which contains the modules "Shell" and "GrfDrv". Actually "GrfDrv" is only required if windows are used, and of course OS-9 Level 2 windows are the key to useful multitasking.

It may be noted that a drive descriptor for an 80 track double sided drive as Drive 0 is not included in the drive selections. Should such a descriptor be needed, we can show you how to make one, or more correctly, we can supply a "patch" for one of the existing descriptors which will convert it to suit.

Once a new "Boot" disk is created, the OS-9 operating system will be running and all hardware in the system will be fully available for the application programmes and multi-tasking that will follow.

The "config" programme calls OS9Gen to create a bootable disk. OS9Gen will write the "Kernel" on track 34 of a floppy disk as well as create the OS9Boot file. Because the Kernel must be in a specific location on the disk, and the OS9Boot must use contiguous sectors, it is best to use a freshly formatted disk as the destination disk for config. Chapter 1 of the "OS9 Commands" section begins with a description of the "Kernel" module and describes other important modules.

Do not expect too much at this point. Remember that OS-9 is not a program, it is an operating system under which many programming languages can be used. Basic09 (a form of Basic), pascal, C, Assembler, Fourth, and Logo to name a few.

Next month we will begin to explore some of the multi-tasking abilities of OS-9. Please let me know if you are having any difficulties in creating a Boot to suit your system.

Gordon Bentzen

oooooooooo0000000000oooooooooo

Additions and Alterations.
By Bob Devries.

Those of you who typed in last months listing of 'CC3Go' could have run into trouble if you did not have the correct definitions file. It was pointed out to me by a local Brisbane user, that not all users out there have the newest version of the definition file 'OS9Defs'. The version I use came from the Development System disk (side B). It has the defs file that was modified for use with OS9 Level II. For those of you who do not have this file, two changes need to be made to the 'CC3Go' listing.
Both changes may be made at the beginning of the file, say at line 7, and you should add these lines:-

```
D.SysPrc equ $4A
D.Crash  equ $6B
```

Please note, that when you assemble this file, you'll get a W warning message at the line that has the 'jmp D.Crash' in it. This is because it is illegal in normal OS9 programming to use a jump instruction or any other direct memory access instructions. In this case, however, OS9 has crashed anyhow, so it no longer matters.

Whilst I am writing about changes to definitions files, there is another one that needs changing. That one is the file 'os9.h' that comes with the C compiler. What I have done, is to add all the necessary definitions for the Level II environment Please note that all these defines are documented in the Operating System Manual. These are the lines I have added to my file:-

```
#define F_NMLINK    0x21        /* CoCo Non mapping Link */
#define F_NMLOAD    0x22        /* CoCo Non mapping Load */
#define F_VIRQ      0x27        /* Install / delete Virtual IRQ */
#define F_GCMDIR    0x52        /* Pack Module Directory */
#define F_ALHRAM    0x53        /* Allocate High Ram Blocks */
#define SS_ALFAS    0x1c        /* Return Alpha Display Status (CoCo SCF GetStat) */
#define SS_CURSR    0x25        /* Cursor Information for CoCo */
#define SS_SCSIZ    0x26        /* Return screen size for CoCo */
#define SS_KYSNS    0x27        /* Get/PutStat for CoCo Keyboard */
#define SS_COMST    0x28        /* Get/PutStat for Baud/Parity   */
#define SS_AAGBF    0x80        /* PutStat to Allocate additional graphic buffer */
#define SS_SLGBF    0x81        /* PutStat to select a different buffer */
#define SS_MPGPB    0x84        /* Get/PutStat to request buffer to be mapped to the workspace */
#define SS_WNSET    0x86        /* Set up high level windowing information */
#define SS_MNSEL    0x87        /* Request high level Window handler to determine next event */
#define SS_SBAR     0x88        /* PutStat to position window scroll bars */
#define SS_MOUSE    0x89        /* Return mouse information packet (CoCo) */
#define SS_MSSIG    0x8a        /* PutStat to tell driver to send signal on mouse event */
#define SS_ASCRN    0x8b        /* Allocate a screen for application poking */
#define SS_DSCRN    0x8c        /* Display a screen allocated by SS_ASCRN */
#define SS_FSCRN    0x8d        /* Free a screen allocated by SS_ASCRN */
#define SS_PSCRN    0x8e        /* Polymorph screen into a different screen type */
#define SS_PALET    0x91        /* Return Palette information */
#define SS_MONTR    0x92        /* Get and Set monitor type */
#define SS_SCTYP    0x93        /* Get screen type information */
#define SS_GIP      0x94        /* Global Input Parameters */
#define SS_UMBAR    0x95        /* Update Menu Bar SetStat */
#define SS_FBRGS    0x96        /* Return Colour registers GetStat */
#define SS_DFPAL    0x97        /* Set/Return default palette registers */
#define SS_TONE     0x98        /* Generate tone using 6-Bit sound */
```

oooooooooo0000000000oooooooooo

Finding information about your files.
By Bob Devries.

For a long time now, I have wanted to find out the details of the data stored in the file descriptor sector of my files, and not just the modified date, starting sector, and length as 'dir e' reports it. Well, I thought that rather than buy a programme, it would be easier and more educational to write my own. So here it is. Type it in, and compile it, and you'll own a very useful tool for your OS9 toolbox.
To use fileinfo, either type 'fileinfo file', where file is any OS9 pathname, or pipe the output of a programme like 'ls' or 'd' to it like this:-

```
ls ! fileinfo -z
```

Bob Devries.

```
/* FileInfo.c - by Bob Devries */
/* Find file header information */
#include <stdio.h>
#include <direct.h>
#include <os9.h>
FILE *fp, *fopen();
struct fildes desc;
main(argc,argv)
int argc;
char *argv[];
```

```
{
    int argptr = 1, i, piped = FALSE;
    long pos;
    long lsn, *ptr;
    char s[11];
    char attr[9];
    char *utoa();
    char *getattr();
    char filename[80];
    pflinit();
    if (argc--<2) {          /* check if the right params are passed */
        puts("Usage: fileinfo file [file] ...");
        puts("   or: fileinfo -z   to use stdin");
        exit(0);
    }
    if ((argv[1][0] == '-') && (toupper(argv[1][1]) == 'Z'))
        piped = TRUE;        /* -z param so use read stdin for filenames */
    do {
        strcpy(attr,"dsewrewr");
        if (piped) {
            if((fgets(filename,80,stdin)) == NULL) { /* read stdin */
                piped = FALSE;
                break;
            }
        filename[strlen(filename)-1] = '\0';     /* remove CR char from name */
        } else {
            strcpy(filename,argv[argptr]);       /* if not stdin use command line params */
        }
        if ((fp = fopen(filename,"r")) == NULL) {     /* try to open filename as a file */
            if((fp = fopen(filename,"d")) == NULL) { /* if that fails, try directory read */
                fprintf(stderr,"\nCan't open %s\n",filename); /* else report failure */
                argptr++;
                continue;
            }
        }
        if ((fseek(fp,0l,2))== -1) /* seek to end of file */
            fprintf(stderr,"Something wrong! Can't find end of file %s.\n",filename);
        pos = ftell(fp); /* get number of chars in file */
        printf("\nInfo on file %s.\n",filename);
        printf("\nOS9 reports that file is %ld ($%lX) bytes long.\n",pos,pos);
        readfd();        /* read file descriptor sector */
        printf("\nFile descriptor information follows :\n\n");
        printf("File attributes are : %s.\n",getattr(attr,desc.fd_att));
        printf("File owner is %s.\n",desc.fd_own==0 ? "SuperUser" : utoa(desc.fd_own,s));
        printf("File was last updated on %02d/%02d/%02d at
%02d:%02d.\n",desc.fd_date[0],desc.fd_date[1],desc.fd_date[2],desc.fd_date[3],desc.fd_date[4]);
/* the previous 2 lines should be typed in as one line */
        printf("File link count is %d.\n",desc.fd_link);
        printf("File size is %ld.\n",desc.fd_fsize);
        printf("File was created on %02d/%02d/%02d.\n",desc.fd_dcr[0], desc.fd_dcr[1],desc.fd_dcr[2]);
        ptr = &lsn;
        for(i=0; i<48; i++) {     /* print segment list */
            l3tol(ptr,desc.fdseg[i].addr,1);
            if(lsn) {
                printf("Used %u %s, starting at LSN %ld ($%lX).\n",desc.fdseg[i].size,(desc.fdseg[i].size > 1) ?
"sectors" : "sector",lsn,lsn);
/* the previous 2 lines should be typed in as one line */
            } else {
                break;
            }
        }
        argptr++;
        fclose(fp);     /* close the file */
    } while ((--argc) || (piped)); /* and continue */
}
char *utoa(number,string) /* unsigned to ascii conversion modified from K & R itoa() function */
unsigned number;
char string[];
{
    int i;
    unsigned sign;
    if ((sign = number) < 0)
        number = -number;
    i = 0;
    do {
        string[i++] = number % 10 + '0';
    } while ((number /= 10) > 0);
    if (sign < 0)
        string[i++] = '-';
    string[i] = '\0';
    reverse(string);
    return(string);
}
reverse(s)
char s[];
{
    int c, i, j;
    for (i=0, j=strlen(s)-1;i<j; i++, j--) {
        c=s[i];
        s[i] = s[j];
        s[j]=c;
    }
}
char *getatir(attr,byte) /* from a attribute byte, return a string with chars set or '-' */
char attr[];             /* e.g. --ewrewr */
char byte;
{
    int i, j, k;
    for (i=0, j=128;i<8;i++,j/=2) { /* must use j/=2 here NOT >> because of sign extension */
        k = byte & j;
        if(!(k))                    /* if bit not set, insert a '-' char */
            attr[i] = '-';
    }
    return(attr);
}
readfd() /* read the file descriptor sector into the structure, using the GETSTT SS_FD call */
{
    struct registers regs;

    regs.rg_a = fileno(fp);  /* a = path number */
    regs.rg_b = SS_FD;       /* b = function number */
    regs.rg_x = &desc;       /* x = address of memory packet */
    regs.rg_y = sizeof(desc);/* y = number of bytes of packet */
    regs.rg_u = 0;
    _os9(I_GETSTT,&regs);    /* didn't check for errors here, if error, OS9 must have crashed */
}
```